# A Multistandard FFT Processor for Wireless System-on-Chip Implementations

Ramesh Chidambaram, Rene van Leuken
Department of Electrical Engineering (CAS group),
Delft University of Technology
The Netherlands
{r.chidambaram, t.g.r.vanleuken}@ewi.tudelft.nl

Marc Quax, Ingolf Held, Jos Huisken
Silicon Hive
Eindhoven, The Netherlands
{marc.quax, ingolf.held, jos.huisken}@philips.com
www.siliconhive.com

***Abstract*--This paper presents a high performance FFT ASIP. The resulting programmable solution is scalable for the order of the FFT and capable of satisfying performance requirements of various OFDM wireless standards. The IEEE 802.15.3a Ultra Wideband OFDM - being the most time critical of these standards because of the computation of a 128-point FFT within 312.5 ns – has been the primary performance target of the scalable ASIP. The resulting ASIP adopts a vectorial Ultra-Long Instruction Word (ULIW) approach. The design decisions are evaluated with regards to processing speed, area and power dissipation.**

## I. INTRODUCTION

This section states the objectives and the design framework is described. Section II presents design choices. Processor architecture, system integration issues and software are subjects of Section III, IV, and V respectively. A characterization of the processor with regard to processing speed, gate count, power dissipation and scalability is shown in VI. Finally, Section VII summarizes the paper.

### *A. Problem description*

The Fast Fourier Transform (FFT) is an integral part of Orthogonal Frequency division multiplexed (OFDM) wireless communication standards – such as 802.11a/g, 802.16, and 802.15.3a – the requirements of which are tabulated in Table I.

TABLE I. FFT PERFORMANCE REQUIREMENT COMPARISON

| OFDM standard | Commercial name | FFT size | Time for computation |
|---|---|---|---|
| IEEE 802.11 | WiFi (WLAN) | 64 | 4 µs |
| IEEE 802.16 | WiMax (WMAN) | 512 | 146 µs |
| IEEE 802.15.3a | WiMedia (WPAN) | 128 | 312.5 ns |

As seen in Table I, the FFT for the Orthogonal Frequency division multiplexed Ultra Wideband (UWB-OFDM) proposal [1] – commercially referred to as WiMedia - for high-speed WPAN systems is the most demanding in terms of time-criticality as it has an available time span of 312.5 ns for complete computation. Commercially available general-purpose programmable DSP processors do not provide the performance to support such time critical solutions. The design methodology of the presented Application Specific Instruction-set Processor (ASIP) – henceforth referred to as the AVISPA-EXP1 – is such that the FFT order is scalable in software, and derivatives of the ASIP for different requirements can also easily be obtained. Hence the fundamental focus of the present work is meeting the demands of the FFT for UWB-OFDM.

### *B. Silicon Hive's ASIP design template*

Silicon Hive has developed a proprietary design methodology and toolset to enable automatic instantiation of an architecture template [2]. All of Silicon Hive's processors are derived from this template, referred to as ULIW (Ultra Long Instruction Word) technology. The basic component of the processors is the processing and storage element (PSE). A PSE is configurable, and a general schematic is depicted in Fig.1.

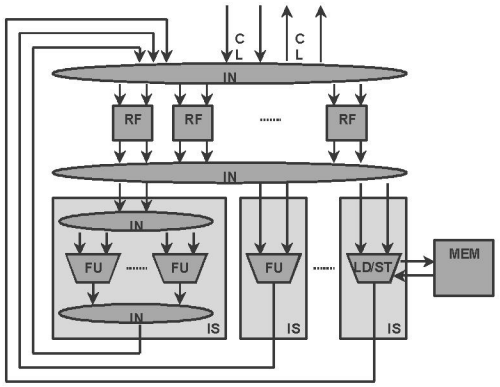


Figure 1. General schematic of a Processing and Storage Element

The PSE is a Ultra Long Instruction Word (ULIW)-like datapath consisting of several interconnect networks (IN), one or more issue slots (IS) with associated functional units (FU), distributed register files (RF) and an optional local memory storage (MEM). The design methodology involves automated instantiation of a processor like the AVISPA-EXP1 from a hardware specification, written in a highly abstract hardware description language called TIM. This enables incorporating rapid changes to the hardware architecture thus easing design iterations for co-simulation.

## II. THE AVISPA-EXP1 PROCESSOR: DESIGN CHOICES

The AVISPA-EXP1 is designed to serve as a target device to execute the twiddle factor driven, decimation in frequency (DIF) Radix-2 in-place algorithm [3]. The order of the FFT (*N*) is made scalable to suit different communication standards. The regularity and direct generic applicability inherent within the radix-2 approach makes it easy to scale as compared to a Split/Mixed radix or other specially tailored approaches. Adopting a radix-2 approach to implementing the 128-point FFT for UWB-OFDM implies that the 448 radix-2 butterflies have to be computed within 312.5 ns. With this requirement, 8 radix-2 butterflies are chosen to operate simultaneously. In a 16-bit data path processor, the warranted highly vectorial approach required presents a memory bandwidth issue. Packing two 16-bit data values - representing real and imaginary components – comprises a complex value. Since 8 butterflies are simultaneously to be fed data *every clock cycle*, it translates into two 8x32 = 256-bits wide inputs to the IS that performs the FFT computation. To provide data and twiddle factors to an IS that computes 8 radix-2 butterflies simultaneously, three 256-bit input ports and two such output ports are required. The design choices to facilitate a stage-wise twiddle factor driven approach using a specialized IS based embedded FFT processor are:

### A. *Dual Ported Data Memories*

Since 8 butterflies are simultaneously to be fed data *every clock cycle*, for such a continuous flow of input data Load/Store Units (LSUs) are used as it meets performance requirements and by nature provides guaranteed data availability for the FFT IS. The second important issue is simultaneously fetching data from different locations, *(N/2)* apart for each stage. This fetching has to take place in a very streamlined manner for effective software pipelining. Using dual ported memories - which also allows data to be stored simultaneously on two distinct locations within a memory - circumvents this issue.

### B. *Twiddle Storage on Processor Memory*

While data values can be read in a vectorial manner, by contrast vectorially loading of 8 twiddle factors is not straightforward. Between stages, the FFT undergoes a size division-by-2. Hence, except for the first stage, the twiddle factors do not reside contiguously in memory. In the present case, 8 butterflies requiring their corresponding twiddle value have to be all procured within a single clock cycle. There are two broad classifications of the approaches to twiddle handling. The first is that of fetching and storing twiddles in the processor memory, i.e. on chip RAM, as part of the processor input/output memory blocks. The other method is to generate the twiddles by calculating them when required using a CORDIC (Coordinate Rotation Digital Computer). In the present case, twiddles are stored in memory. The 8 radix-2 FFT butterflies operating in parallel would require 8 dedicated CORDICs generating 16-bit precision values, thereby translating into much larger area on chip as compared to storing twiddle factors in memory [4]. Hence it is not preferred.

### C. *Data and Twiddle Factor Ping-ponging*

The idea of software pipelining is that the body of a loop can be reformed so that a loop iteration can start before previous iterations finish executing. Software pipelining of loop operations is highly preferred owing to the advantages that accompany a static ULIW instruction word. Besides allowing a high extent of parallelism and reduced cycle-count, there is a significant power saving in turning off the program memory when the instruction word remains unchanged between cycles. In order to pipeline the three fundamental loop operations – i.e. data fetching from memory, FFT butterfly computation, and data storing to memory – to a single cycle loop, memory ping ponging is used. The concept of memory ping ponging is to have a separate source memory from which data is read and a separate destination memory to which data is written after computation. This overcomes the memory bandwidth and contention issue. The source and destination memories are interchanged between stages. A pictorial representation of memory ping ponging is depicted in Fig.2.

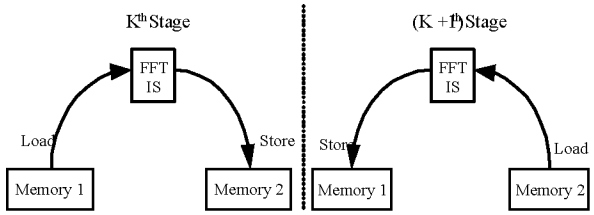


Figure 2. Concept of memory ping ponging

In order to be able to procure all the required twiddle factors in a single load operation (because, in a given stage they all reside on the same memory) a separate IS in the data path is used to perform selection of alternate twiddle factors and store them contiguously into another memory, as depicted in Fig.3.

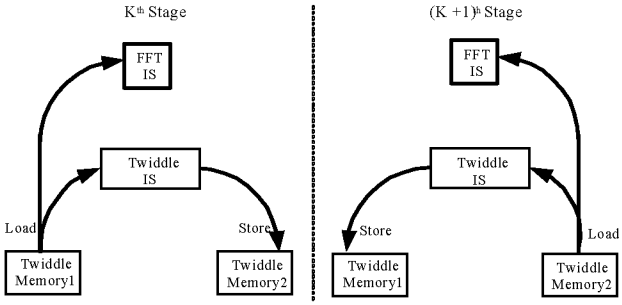


Figure 3. Concept of twiddle factor ping-ponging

### D. *Instruction Set of the FFT FU*

The AVISPA-EXP1 houses a specialized vectorial FFT FU, which calculates 8 radix-2 butterflies in tandem. In order to retain accuracy, the data values are shifted to the right by one bit, every alternate stage [5]. Thus the instruction set of the FFT FU has separate instructions for even and odd stages of the FFT. During even stages, the data values from the outputs of the butterfly computation undergo an arithmetic shift right by one bit. Besides the general case, wherein the butterflies require data derived from *both* the input vectors, the last three stages undergo

intra-vectorial computation. These last three stages are realized by incorporating six corresponding instructions – two instructions per either an even or odd stage occurrence. Hence, including the general inter-vectorial stage instructions, the FFT FU has 8 enabling instructions.

With an approach of halving data every even stage, a 16-bit resolution yields an average signal to quantisation noise ratio (SQNR) of 75 dB for the 128-point FFT.

## III. The AVISPA-EXP1 Processor: Architecture

The architecture of the AVISPA-EXP1 processor is shown in Fig.4. It contains one FFT-specific PSE and one generic PSE (GP) for typical ALU operations and handling of initialization of LSUs within the FFT PSE. The GP PSE ensures C-programmability of the processor. It contains a branch controller combined with status update unit; a send & receive unit to exchange tokens with external devices; two 32-bit load/store units connected to local memory; two ALU units; two modulo-add units and two shift units. The GP PSE executes control and data transfer tasks as well as synchronization with other SoC blocks.

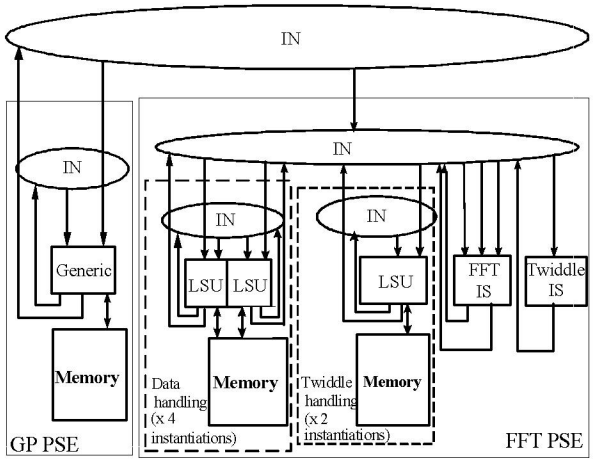


Figure 4. Architecture of AVISPA-EXP1 processor

The FFT PSE interfaces with 4 dual ported data memories and 2 single ported twiddle factor memories. The former requires 8 256-bit LSUs and the latter requires 2 256-bit LSUs. To procure and dispatch data as input to the first stage and output of the last stage, 2 separate dual ported memories are used with dedicated LSUs. The data path for twiddle handling is kept isolated from the data path for data values. Hence, a total of 12 IS reside within the FFT PSE. A schematic view of the resulting architecture functionality is depicted in Fig.5.

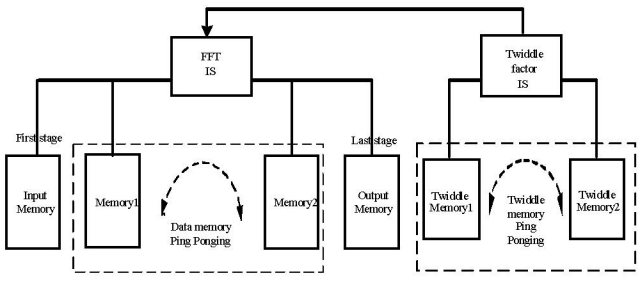


Figure 5. Schematic view of FFT PSE

## IV. System Integration of the AVISPA-EXP1

The FFT processor typically forms a component of a SoC performing, for instance, demodulation according to the UWB-OFDM standard. Therefore, the processor provides communication interfaces and executes a communication protocol to synchronize with other SoC components. There are two communication interfaces to communicate with external devices. The first interface is a streaming interface, used for FIFO-based communication with adjacent processing blocks. The stream interface enables a token-based synchronization protocol that handles timely exchange of processed data to avoid overall system stall. The second interface is a high-speed data bus. The data bus is used for block-based data communication to the previous and next block in the processing chain. The data transfer is controlled by the synchronization protocol.

## V. The AVISPA-EXP1 Processor: Software

Given the overhead inherent with twiddle factor handling in a highly vectorial processor, adopting a twiddle-factor driven approach to writing the application kernel is justified. Each twiddle set (of 8 twiddles) is loaded just once per stage, and all the butterflies corresponding to those twiddles are computed before loading the next twiddle set. The implementation of such a scheme takes the shape of a loop-in-loop. The innermost loop is restricted to only butterfly computation for a particular twiddle factor set, while the outer loop deals with handling twiddle factors. Three primary operations of loading the next input, computing the FFT butterflies, and storing previous output are software pipelined to within a single clock cycle. The alternate usage of odd and even stage instructions is facilitated by an outer global loop, which consequently also controls inter-stage memory ping ponging. The pseudo-code for the general case is as shown below:

```
Loop1 : for (stage_i = 1 to (log2 (N) – 3) )
  Loop2: for(twiddle_i = 1 to T_STAGE)
    Loop3: for(data_i = 1 to N_STAGE/2)
      Load_Mem1(data_i, data_i + N_STAGE/2);
      Odd_BF(data_i, data_i + N_STAGE/2, twiddle_i);
      Store_Mem2(data_i, data_i + N_STAGE/2);
    End; -- loop3
  End; -- loop2
Recalculate_stage_parameters(N_STAGE, T_STAGE );
Check_for_exit_loop1( );
  Loop4: for(twiddle_i = 1 to T_STAGE)
    Loop5: for(data_i = 1 to N_STAGE/2)
      Load_Mem2(data_i, data_i + N_STAGE/2);
      Even_BF(data_i, data_i + N_STAGE/2, twiddle_i);
      Store_Mem1(data_i, data_i + N_STAGE/2);
    End; -- loop5
  End; -- loop4
End; -- loop1
```

The first stage (since it deals with separate Input Memory) and last three stages (as they involve intra-vectorial computation) are special cases and hence handled outside of this global outer loop.

## VI. AVISPA-EXP1 PROCESSOR: CHARACTERIZATION

The AVISPA-EXP1 can be characterized by the metrics processing speed, gate count and power dissipation. The extent of scalability in cycle count with increase FFT order is examined. The extent of changes to be incorporated in hardware for handling varying FFT sizes is minimal and if needed, only in terms of memory availability. But if required, derivatives of the AVISPA-EXP1 for different extents of parallelism can also easily be obtained.

### A. *Processing Speed*

The AVISPA-EXP1 was designed and verified for correct operation. The data throughput rate attains UWB-OFDM specifications, at a clock speed of 336 MHz.

### B. *Gate Count*

In CMOS 0.12µm technology, the AVISPA-EXP1's core was estimated at a gate count of 365 kGates and a core size of 2.4 $mm^2$ (excluding memories). A comparison with the FFT module within an ASIC solution [6] for UWB-OFDM is illustrated in Table II. The FFT block in [6] comprises Radix-8/4/2 mixed-radix solution butterflies, while the core of the AVISPA-EXP1 consists of 8 Radix-2 butterflies.

TABLE II. AREA COMPARISON WITH ASIC SOLUTION

| Design | Process (nm CMOS) | Area of FFT module ($mm^2$) |
|---|---|---|
| [6] | 180 | 4.22 |
| AVISPA-EXP1 | 120 | 2.44 |

### C. *Power Dissipation*

Given an average hardware utilization of approximately 30%, the core alone is estimated to dissipate 188 mW at 336 MHz clock speed in CMOS 0.12µm. Taking the memories also into consideration – which have been designed to accommodate a limit of a 1024-point FFT - the power dissipation is 1.78 Watts.

### D. *Scalability in Software*

Besides attaining UWB OFDM standard's time-criticality requirements, the AVISPA-EXP1 is designed having ease of scalability as a primary objective. The application kernel was compiled and scheduled for different FFT sizes. The cycle counts for the different FFT sizes is enumerated in Table III. The ULIW instruction width is 619-bits, and the program length is 152 instructions long.

TABLE III. SCALABILITY IN TERMS OF NUMBER OF CYCLES

| Order of FFT (N) | Cycle count |
|---|---|
| 64 | 54 |
| 128 | 105 |
| 256 | 216 |
| 1024 | 1380 |

For the cases of larger N, the extent of relative overhead with respect to ideal cycle count (case of zero overhead) decreases. The solution in [7], which uses a reconfigurable cell based technology, and the AVISPA-EXP1 are juxtaposed in Table IV with regard to various parameters. As can be noted, the AVISPA-EXP1 outperforms the FFT implementation in [7] in all aspects. Present area and power figures assume a 1024-point FFT, which stems from memory size limitations.

TABLE IV. COMPARISON WITH A SCALABLE FFT PROCESSOR

| Characteristic Aspect | [7] (130nm, 450 MHz) | AVISPA-EXP1 (120 nm, 336 MHz) |
|---|---|---|
| Performance: For *N*=128 FFT, | | |
| ▪ Cycle Count | 225 | 105 |
| ▪ Time period | 500 ns | 312.5 ns |
| Scalability: Cycle count for: | | |
| *N*=64 | 111 | 54 |
| *N*=128 | 225 | 105 |
| *N*=256 | 520 | 216 |
| *N*=1024 | 2613 | 1380 |
| Peak power (core + memories) | 4 Watts | 1.78 Watts |
| Area (Without memory) | (8 x 8) $mm^2$ | 2.44 $mm^2$ |

## VII. SUMMARY

The AVISPA-EXP1 is a 'big step' in attaining a level of performance that is comparable to an ASIC solution. The AVISPA-EXP1 is a highly parallel, vectorial architecture. A specialized FFT hardware unit that computes 8 radix-2 butterflies simultaneously is used. The twiddle factor driven approach has been chosen and the application kernel is fully scalable for the size of the FFT. Although the area is dominated by the processor memories, the area of the FFT core is estimated at 2.44$mm^2$, which proves comparable with an ASIC solution, which uses a mixed radix approach. The extent of changes to be incorporated in hardware for handling varying FFT sizes is minimal and only in terms of memory availability. Scalability has been addressed by providing scheduled cycle counts for different FFT cases and compared against another rival solution.

## REFERENCES


[1] A. Batra, J. Balakrishnan and A. Dabak, "Multi-band OFDM: a new approach for UWB", Proc. ISCAS,Vol. 5,pp. V365-V368, May 2004

[2] T.R. Halfhill, "Silicon Hive Breaks Out", Microprocessor Report, Dec 2003, Available at www.MPROnline.com and www.siliconhive.com

[3] Y. Jiang, T. Zhou, Y. Tang and Y. Wang, "Twiddle-factor-based FFT algorithm with reduced memory access", Proc. IPDPS., pp.70-77, April 2002

[4] A. Madisetti,A. Y. Kwentus and A. N. Willson Jr., "A 100-MHz, 16-b, direct digital frequency synthesizer with a 100-dBc spurious-free dynamic range", Trans. Solid-State Circuits, Vol. 34, Issue 8, pp. 1034 - 1043, Aug. 1999

[5] J. Armstrong, H. A. Suraweera, S. Brewer and R. Slaviero, "Effect of rounding and saturation in fixed-point DSP implementation of IFFT and FFT for OFDM applications", Proc. The Embedded Signal Processing Conference (GSPx 2004), Sep. 2004.

[6] H. Y. Liu, et al, "A 480Mb/s LDPC-COFDM-Based UWB baseband transceiver", Proc. ISSCC. San Francisco, Feb. 2005

[7] A.H. Kamalizad, C. Pan, N. Bagherzadeh, "Fast parallel FFT on a reconfigurable computation platform", Proc. Computer Architecture and High Performance Computing, pp. 254 - 259, Nov.2003